One step forward: Linking Wireless Self-Organizing Networks Validation Techniques with Formal Testing approaches

Stephane Maag, Aline Carneiro Viana, Fatiha Zaïdi

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

One step forward: Linking Wireless Self-Organizing Networks Validation Techniques with Formal Testing approaches

Stephane Maag — Aline Carneiro Viana — Fatiha Zaidi

N° 6817

January 2009

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--6817--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY - ÎLE-DE-FRANCE

One step forward: Linking Wireless Self-Organizing Networks Validation Techniques with Formal Testing approaches

Stephane Maag* , Aline Carneiro Viana , Fatiha Zaidi†

Thème COM — Systèmes communicants
Équipes-Projets Asap

Rapport de recherche n° 6817 — January 2009 — 43 pages

Abstract:

Multi-Hop Wireless Self-Organizing Networks (WSONs) have attracted considerable attention from the network research community; however, the key for their success is the rigorous validation of the properties of the network protocols. In particular, applications of risk or that demand precision require a rigorous and reliable validation of deployed network protocols. That is the reason why many efforts have been performed in order to validate the requirements and the functioning of protocols in such kinds of networks. It can be observed, however, that, even if different communities have carried out intensive research activities on the validation domain, WSONs still raise new issues and challenging constraints to these communities. The goal of this tutorial is to present a comprehensive review of the literature on protocol engineering techniques and to discuss difficulties imposed by the characteristics of WSONs to the protocol engineering community.

Key-words: survey, protocol engineering, multi-hop self-organizing networks, modeling techniques, performance of systems, simulation

* TELECOM & Management Sud Paris
† University of Paris Sud

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université
4, rue Jacques Monod, 91893 ORSAY Cedex
Téléphone : +33 1 72 92 59 00

Plaidoyer pour un rapprochement entre les approches formelles et non formelles pour la validation des réseaux auto-organisables

Résumé :

Les réseaux auto-organisants à sauts multiples ont suscités un intérêt grandissant de la communauté réseau, cependant la clé de leurs succès réside dans une validation bien établie des propriétés des protocoles de ces réseaux. En particulier, les applications liées à la sécurité ou qui demandent une grande fiabilité nécessitent une validation rigoureuse et sûre des protocoles qui sont déployés sur le réseau. Cette nécessité explique pourquoi beaucoup d'efforts ont été réalisés afin de valider les exigences et le fonctionnement des protocoles de tels réseaux. On peut noter que en dépit de l'intense activité des différentes communautés de recherche dans le domaine de la validation, les réseaux mobiles auto-organisants soulèvent encore des problèmes ouverts et des contraintes à considèrer. L'objectif de ce tutoriel est de présenter de façon comprehensible une revue de la littérature sur les techniques d'ingénierie des protocoles et de discuter des difficultés qu'induisent les caractéristiques spécifiques des réseaux mobiles auto-organisants et qui s'imposent à la communauté de l'ingénierie des protocoles.

Mots-clés : tutoriel, ingénierie des protocoles, réseaux auto-organisants à sauts multiples, simulation

1 Introduction

Context. "It is not the biggest nor the fastest of the species that survive, but the one that adapts to its environment" (by Charles Darwin, Theory of Evolution, 1809-1882). Nature is full of interesting examples of systems with self-* (self-configuration, self-organization, etc.) properties, constituting a valuable source of inspiration for the engineering of fully autonomous formation of networks. In addition, advances in communication technologies and the proliferation of wireless computing and communication devices are opening new ways for mobile users to get connected to each other. As a consequence, autonomic networks have emerged with the goal of relying on processes of evolution, development, self-organization, adaptation, learning, teaching, and goal orientation. This futurist goal can be represented by the design of multi-hop wireless self-organizing networks (WSONs) that are able to robustly respond to dynamically changing environments, operating conditions, and purposes or practices of use; thus, facilitating new ways to perform network control, management, and service creation. Wireless networks such as sensor networks, mesh networks, vehicular networks, delay tolerant networks, and MANETs are some examples of networks that follow the principle of WSONs.

Over the last number of years, multi-hop wireless networking area has thus attracted considerable attention within both industry and academia. One reason for this popularity is for sure, the wide range of novel applications in the areas of health, military, environment, and home. The requirements of such applications have, however, a direct impact on the design of the wireless network. In military areas, for instance, rapid deployment, self-organization, and fault tolerance characteristics should be assured. In environmental areas, reliability, fault tolerance, and robustness are important issues, and constitute fundamental characteristics, for instance, in alert-based monitoring applications.

Hence, it can be easily concluded that the success/quality of those applications is then strongly related to the correctness and good performance of the involved network protocols. In particular, safety-critical applications (like healthcare-related or alert-based systems) require a rigorous and reliable validation of all network functionalities and features[1] In addition to threaten people's lives, faulty software also costs money. The fact that people rely on computers in practically every aspect of their lives (e.g., in cars, ATMs, cell phones, etc.) makes higher the cost of unreliable design [[62]].

Motivation. In this way, many efforts have been performed in order to validate the requirements and the functioning of protocols in such kind of networks. While the main goal is to ensure the reliability of the protocols, validation techniques also allow the establishment of their correctness regarding the related requirements. In particular, the properties to be validated are related to behavioral aspects, which are commonly known as *functional* (*e.g.*, protocol interactions, or loop free) and *non-functional* properties (*e.g.*, performance-related issues, like latency, delivery ratio, etc.). In this way, validation techniques have been studied by the research community through different approaches. In particular, functional and/or non-functional properties have been validated by the use of *formal* or *non-formal* approaches.

In the multi-hop wireless networking area, the major techniques used to design and ensure the quality of the network-related protocols essentially rely on descriptions for simulation and/or emulations, even if some works put also trust in mathematical

[1] Here, reliability means that all the application's behaviors are correct against all specified criteria.

models for the understanding of systems' behavior. More specifically, the majority of works rely on non-formal models provided as input to simulators such as NS-2 [[100]], OPNET [[102]], or GloMoSim [[55]]. In this case, simulation is usually conceived to observe and analyze the protocol performance. Nevertheless, works in the literature [[23]; [79]; [6]] have shown that there are growing concerns regarding the reliability of results generated by wireless network simulators. In addition, they have also mentioned the scarcity of results gotten from real experiments [[4]] and the huge diversity of results from simulation when compared to the ones from real case studies. Otherwise, even if emulation testing [[136]; [146]] comes closer to the reality, the simulation test is still required and represents an important component in the emulation testing. Hence, the combination of simulation and emulation techniques is not still enough to replace a real case study [[15]]. Finally, although useful for performance evaluations of protocols, such techniques do not allow one discerning design errors or defining automation of well-defined processes, important issues for evaluating the functional behaviors of protocols.

Recently, some works in the literature have then advocated the use of formal models to test WSONs routing protocols [[44]; [46]; [45]], as a way to deal with the previously described constraints of non-formal models. *Verification* and *testing* are two complementary stepwise techniques for formal protocol validation. The verification technique consists in a formal modeling of the protocol in order to verify some of its properties.[2] Otherwise, testing techniques work on implementations rather than models. In this way, test sequences generated from the formal model are injected in to the final implementation of the protocol. This will allow the comparison between the real results and the expected results provided by the specification.

Nevertheless, formal description techniques and their testing tools have not frequently been applied in multi-hop wireless networking area. This is mainly due to the difficulties that characteristics of WSONs impose to the formal modeling [[145]; [46]]. In particular, as later discussed in this paper, WSONs present a number of characteristics that set them apart from traditional wired networks, as the network dynamicity or the inherently broadcast communication. Thus, even if different communities have carried out intensive research activities on the validation domain, WSONs still raise new issues and challenging constraints to these communities. One example is the scalability issue. In the validation-related works concerning WSONs, the considered network size remains small (e.g., 5 nodes in [[34]], or 18 nodes in [[46]]). This is due to the dynamicity imposed by WSONs, which highly increases the number of states to be considered in the validation process.

Contribution. According to the literature and similarly to researches on the validation area of wired networks, it has been well established that the validation of WSONs protocols may not be addressed by only one method, i.e., formal or non-formal [[45]]. This specifically suggests the integration of formal and non-formal approaches, which still constitutes an open issue in the protocol validation domain.

Following this assumption, it can be observed that similarities can be established between formal and non-formal approaches in terms of techniques and properties to be validated. In this way, we argue that the use of complementary techniques coming from different research communities can help to efficiently address the new constraints imposed by WSONs.

[2] In routing protocols, formal models may check the loop free property of established routes or still the rapid convergence of routes changes.

The goal of this survey is to present a comprehensive review of the literature on protocol engineering techniques and to discuss the challenges imposed by WSONs to the protocol engineering community. Our aim is to provide a better understanding of the current research issues in this field. Following the formal and non-formal classification of techniques, we overview protocol validation approaches, investigate their pertaining design features and constraints, and provide discussion about their similarities. We also investigate how to take advantage of such similarities to obtain complementary techniques.

The scope of the work presented in this paper is distinguished in many aspects from existing surveys on validation techniques [[30]; [64]]. In particular, our work differs from other surveys as follows:

- In general, the literature presents a collection of validation-related works that are adapted to a particular type of problem (*e.g.*, leader election, or a specific routing protocol). Instead, our goal is to help the reader understanding the *foundations* of validation techniques. In this way, we survey both formal and non-formal validation approaches and provide discussion about their limitations and drawbacks. We then, examine the attempts of convergence addressed in the literature.
- The related surveys in the literature are devoted to wired networks only. Due to the importance of WSON and its new challenging characteristics, a detailed discussion on the changes introduced to protocol engineering domain becomes necessary and useful at this stage. Thus, our work is also a dedicated study of particularities introduced by multi-hop wireless networks. In addition, we investigate how validation techniques have been rethought to allow the application to WSONs.
- Finally, we discuss open research problems. We believe the provision of more general insights across the validation techniques, is an interesting direction through the design of novel validation techniques adapted for WSONs.

Outline. The reminder of this survey is organized as follows. We start our analysis by providing in Section 2 an overview of the protocol engineering domain. In Section 3, we discuss the new challenges introduced by the wireless self-organizing networks and that impose limitations to the existent validation techniques. In Section 4 and 5, we examine in detail the formal and non-formal approaches used in the protocol engineering domain. We then survey concepts and discuss the methods used in each approach. Section 6 provides a discussion about the advantages and drawbacks of those approaches regarding the WSONs validation. Section 7 examine the demands for convergence between formal and non-formal domains and discuss the interest of this convergence to deal with the characteristics of WSONs. Finally, Section 8 summarizes our investigations and presents our conclusions.

2 Protocol engineering

The protocol engineering domain covers a large range of activities, from the requirement specification to the final deployment, passing through the design phase. There exists in the literature several way to perform the development of a protocol. All these processes of development share a common starting point.

The design and the validation of a protocol is preceded by the specification of its rules and format of messages. Messages allow the establishment of communication between entities in a distributed system. In this way, once primitives, data units, and communication rules are defined, *protocols lifecycle* can be then started. This consists in the execution of three main phases: the *development*, the *exploitation* (i.e. final deployment), and the *maintenance* (see Figure 1). This paper addresses the development phase, important to ensure the correct design of a protocol to be deployed and maintained in a real environment. The development phase can be proceed according to several models, Waterfall [[119]], such as V-model [[141]], spiral [[17]], prototyping, RAD [[94]], and RUP [[67]] models. More specifically, protocol lifecycles can be divided into two mains classes: the linear and the iterative lifecycles. The waterfall and the V-model are known as linear lifecycles. The name Waterfall [[119]] derives from the cascading effect existent between the steps. When this model is used, no intermediate evaluation is performed between the starting point of the project and the validation step. Hence, this lack of intermediate evaluation has as consequence the fact that no way to follow the development process exists and then no means to organize a work within a team is available. Such a model increases the risk of having errors in the system due to the late validation step. Although being the least flexible, the Waterfall model is well suited for projects with few participants and when the risks are well determined from the starting point of the project.

The second linear model, the V-model [[141]], where formal methods are used, is stated as the most famous model, being experienced in big projects. For these reasons, we selected the V-model as the model of the development phase of a protocol lifecycle to be well detailed in the following section. The main advantage of such a model relies on a model driven by documentation which is produced at every step of the protocol lifecycle. The risks of building a wrong system are also reduced as a validation is performed at each step. The weaknesses are also related to the documentation when it becomes more undershot on the integration. The intermediate validations can not prevent the transmission of deficiency of the previous steps of the lifecycle.

In the family of iterative models, we can cite the prototyping and spiral models [[17]]. The former model is a cyclic version of the linear model. In this model, once the requirement analysis is done and the design for a prototype is made, the development process gets started. Once the prototype is created, it is given to the customer for evaluation. Their response creates the next level of requirements and defines the next iteration. The main strengths are that the users can see steady progress, the feasibility of a proposed design approach can be examined, and the system performances issues can be then explored. The drawbacks rely on the possibility for the users to treat the prototype as the solution and on the fact that a prototype is only a partial specification. Moreover, with such a model, there is no way to know the number of iterations that will be required. On the other hand, the spiral mode consists of a risk reduction by breaking a system project into mini-projects, each one addressing one or more major risks. After major risks have been addressed, the spiral model ends up as a waterfall model. The strengths are on the early iterations, which are the cheapest and enable the higher risks to be addressed at the lowest total cost. In this way, the iterations can be tailored in an efficient way to fit the project's needs. The weaknesses are on the ability to make it in practice as it requires attentive management knowledge.

Other models exist such as: the RAD model (Rapid Application Development) [[94]], which is a linear sequential model that emphasizes an extremely short development cycle that relies on a component-based construction, and the RUP model (Rational Unified Process) [[67]], which is related to the spiral model and has an underlying

object oriented model. We can also mention the agile methodologies which are based on an iterative process. Interested readers can refer to [[70]].

To conclude, no lifecycle model is perfect and the choice of the right model is really dependant on the system to be developed and on the trade-off to be found between all the parameters considered in the system development phase, such as the size project, the development team size, etc. (see [[141]], the chapter 3 for a good overview of lifecycle models). As stated before, we describe in the following the V-model in detail, where the development is performed in several steps, having at each step, a validation process and an associated technical reports. These steps are shared by several lifecycle models.

Protocol development phase:

A protocol development cycle is divided into several steps, which are detailed in this section. The first step captures the requirements of the user or application in terms of available and requested services. This constitutes in a high level task, since no specification of how the system internally works is required. Instead, this first step specifies how the system reacts to interactions coming from the environment. Traditionally and especially in the formal-related area, the requirements are expressed by sequence diagrams, called the *Message Sequence Charts* [[66]], which represent the exchanges of interactions between the different entities of the system and its environment.

The second step designs the protocol in terms of data structures and data units exchanged to manage the timing constraints (i.e. the management of timers). More specifically, the design of a protocol relies on the available service to supply the requested service. The protocol design can be performed by means of formal approaches, as system modeling methods or Formal Description Techniques (FDT), or directly by non-formal description techniques.

A FDT builds a formal model of the protocol that can be used for several purposes, such as formal verification, formal validation of users' requirements, as well as to generate the tests to be executed on the real implementation. The use of a FDT allows checking the correctness of the protocol regarding its expected behavior (see Section 5).

Figure 1: Protocol lifecycle.

Verification and validation (or testing) are complementary techniques for formal protocol description. At the verification step, properties related to the service and those related to the protocol can be verified. If the service-related properties are verified on the model, it can be then established that the model is correct regarding what the system is expected to do, e.g. if a route between two nodes can be correctly established. Protocol-related properties are verified on the model in order to establish that the model is free of deadlocks, livelocks, etc. After the protocol verification is concluded (i.e. it is correct and corresponds to the specified requirements) then, from the correct model, test cases that cover test objectives can be automatically generated.

Finally, the V-model cycle of a protocol is finished by its implementation. If a formal description was adopted in the previous steps, the implementation can be directly generated from the formal model. There exists many code generators from formal models, however, the code generated is not complete and needs to be finalized. The tests automatically generated from the formal model can then be exercised on the final implementation to check whether the implementation conforms to its specification.

Performance analyzers are actuated after the design phase. With formal approaches (such as the system modeling methods described in Section 5.1) or with non-formal approaches (such as simulation, emulation, real-live testing, and prototyping, described in Section 4), the issue to be addressed is the performance evaluation of the developed protocol. In the realm of performance evaluation, the previous steps of requirement and design specifications, are also important steps to be performed. Nevertheless, the requirements are expressed by means of scenarios directly produced from the informal description of the protocol. In the case of some non-formal approaches, for example, these scenarios are scripts to be executed on the simulated or emulated protocol. The protocol is then implemented inside the simulator/emulator according to the data structures and algorithms established at the design phase.

In summary, the design phase of the protocol engineering domain is addressed in this paper, by two different approaches: the formal and the non-formal. Approaches based on well founded semantic, such as methods based on formal description techniques and on system modeling methods are studied in this paper as formal approaches. Otherwise, techniques like simulation, emulation, real world testing, and prototyping are studied as non-formal approaches. Formal and non-formal approaches are complementary techniques to be used at the development of new protocols.

Figure 2: General classification of protocol design approaches.

Figure 2 presents the general taxonomy of the protocol development approaches. A survey on non-formal and formal approaches as well as a discussion of their strengths and weaknesses are provided at the following sections. Moreover, in order to better investigate their characteristics when compared to their use in wired network area, the next section first describes the new challenges imposed by WSONs that affect the way protocols should be designed.

3 The WSONs challenges

Wireless self-organizing networks introduce new challenges to the project engineering research community. This is due the fact they present radically different technical characteristics that set them apart from wired networks. In the following, we describe the particularities of these networks. Section 6 and 7 then discuss the difficulties imposed by the new challenges of WSONs and how validation techniques have been rethought to be applied in WSONs.

- *Broadcast communication.* – Unlike in wired network, where the point-to-point model of communication is dominant, communication in wireless networks is inherently broadcast based. That is, when a node transmits some information, typically all nodes within its transmission range can receive it. The broadcast nature of wireless communication can be exploited in a number of applications, including information dissemination.

- *Communication paths determined by physical location* – Due to the wireless link properties of WSONs, neighborhood is imposed by the physical location of nodes. Nodes should then rely on their physical immediate neighbors for communication.

- *Links are unreliable* – Variables such as obstructions, interference, environmental factors, and mobility make determining connectivity a priori difficult in WSONs. Also, contrary to wired networks, in which the channel can be characterized reasonably well, much more unpredictability is expected in the wireless case. Thus, the low link reliability and its possible asymmetry require that protocol designed for WSON be fault tolerant and adaptable to connectivity changes.

- *Collision* – Contrary to wired networks, collision detection is not feasible in WSONs and collision avoidance is hard to achieve. Thus, links may suffer a much higher percentage of message loss through collisions.

- *High heterogeneity* – In WSONs, it is likely that nodes are heterogeneous in their characteristics such as memory availability, computation capacity, and transmitting power.

- *Potentially high mobility* – Due to the absence of wiring, nodes in WSONs can be mobile. It has been shown in the literature that nodes mobility is favorable to the spread and/or aggregation of information through the network [[58]; [13]; [139]]. On the other hand, node mobility also imposes changes to the topology. This dynamic topology, in turn, implies more complex management algorithms for topology maintenance and routing. For instance, the dynamic nature causes routes to be unstable and make routing a resource-greedy operation. In this way, in order to have flexibility in route selection and simpler

dynamic-network management, the designed addressing structure and forwarding mechanisms should be as flexible as possible.

- *Constrained resources* – Due to the absence of wiring and their small physical size, wireless devices often have limitations in memory, processing, and above-all, power. In this case, the optimization of resources is strongly required in order to minimize energy consumption and communication overhead. This also requires (1) to take local-scoped decisions through simple neighborhood consensus, and (2) the distribution of information and management responsibilities among the nodes in the network.

- *Vulnerability* – Due to its infrastructure-free and non-authority capabilities, WSONs are inherently insecure. In addition, transmissions are generally in broadcast mode, which makes traffic overhearing easier for any node.

4 Non-formal approaches

In the network community, the major techniques used to debug and evaluate designed network protocols rely on non-formal approaches such as: simulation and emulation tools as well as live-testing experiments (real testbeds) and rapid prototyping. In the wireless network domain in particular, the advantages in hardware development have made possible the deployment of inexpensive, autonomous, and compact sensor devices, which has improved the viability of deploying live-testing experiments of wireless sensor networks [[149]].

Otherwise, the choice of a simulator or an emulator is up to the designer. If having a high view of one idea is enough to evaluate its performance (e.g., reliability of routing protocol, zone coverage guarantees, etc.), the simulation will be the more useful tool. If, however, a fine-tuning precision of low level results is required (e.g. precise timing analysis of the simulated software, etc.), then the emulation will be the more effective tool.

It the following, we discuss each of these widely employed non-formal approaches. Notice, however, that our goal here is to provide discussion about the concept and characteristics of non-formal approaches, instead of surveying techniques existent in the literature for each approach. Although discussions to be focused on sensor networks, a good survey on simulation tools can be found in [[30]].

4.1 Simulation

Simulation plays a valuable role in network research, allowing designers to test networking protocols in a controlled and reproducible environment. Simulations are often used as an adjunct to, or substitution for, modeling systems for which simple closed form analytic solutions are not possible. In the wireless network domain, simulation constitutes an important tool, since the evaluation in real environment of wireless applications requires non-negligible programming skills and time, besides imposing limitations on the network size[3]. Researchers generally use simulation to analyze system performance (i.e. quantitative analysis) prior to physical design, or to compare multiple alternatives over a wide range of conditions.

[3] Wireless scenarios may be particularly difficult or expensive to emulate using real hardware.

Simulation can be defined as *"the representation of the behavior or characteristics of one system through the use of another system, esp. a computer program designed for the purpose."* (according to `dictionary.com` website). For instance, computer programs can simulate weather conditions, chemical and atomic reactions, etc. Theoretically speaking, a computer simulation of a phenomenon becomes possible if mathematical data and equations can be designed to represent it. Nevertheless, the fact that most natural phenomena are subject to an almost infinite number of influences, makes their simulation an extremely difficult task in practice. Therefore, simulations are usually performed by implementing only the most important factors of a phenomenon.

Simulations are also used to test new theories, and in the context of communication and computer network research, new protocols. In the case of network protocols, after creating a theory of causal relationships or a description of interactions between the different network entities (hosts/routers, data links, packets, etc.), a network engineer can codify these interactions in the form of a computer program. This program or simulator models the behavior of a network either by calculating the interaction between the different network entities using purely mathematical models, or by capturing and playing back observations from the behavior of these network entities. If the simulated program behaves in the same way as the real case, there is a good chance that the proposed relationships/descriptions of interactions are correct.

Figure 3: Work-flow describing the steps related to a protocol design based on the simulation non-formal approach.

A workflow describing the steps required in the protocol design process using the simulation non-formal approach is shown in Figure 3. After the protocol description (represented by "idea" in the workflow) the network scenario description must be provided. Considering the case of wireless network simulators, the "scenario description" specifies: the topology (number of nodes, node distribution, network area size), the connectivity (the communication range, the neighborhood density), the network dynamicity (static, nodes join/leave, link failures, mobility model – individual or group mobility), and the traffic characteristics (messages types and format, traffic type, communication model – broadcast, unicast, gossip, etc.). The "implementation" of the protocol specifications is then followed by the experiment description. This latter step has as goal to describe the metrics (delay, delivery ratio, latency, etc.) to be evaluated and consequently, results in the implementation of statistic monitors. These monitors are entities that check the performance metrics during the simulation execution and generate the trace files accordingly. Trace files are generated according to the designer needs and register monitored statistics, which will be used for final protocol analysis. Important examinations should be performed in order to guarantee the correct protocol implementation and scenario specification [[79]].

Typical examples of current used networking simulators are: NS-2 [[100]], OPNET [[102]], GloMoSim [[55]], SimReal [[128]], SENSE [[125]], TOSSIM [[136]], WSNet [[147]], SensorSim [[126]], J-Sim [[68]], SENS [[124]], EmStar [[42]; [53]; [52]], and REAL [[72]]. These simulators target a higher range of protocols and provide a simulation language with network protocol libraries. Instead, the "do-it-yourself" class of simulators are much focused implementations that usually model only the details relevant to the developer.

Most network simulators [[100]; [102]; [55]; [147]; [136]] use discrete event simulation, in which a list of pending "events" is stored, and those events are processed in order, with some events triggering future events – such as the event of the depart of a packet at one node triggering the event of the arrival of that packet at a downstream node. Some others are classified as application-oriented simulators [[124]; [42]] and provide a framework for developing applications on wireless sensor networks. Most systems have also improved programming environment with Graphical User Interface (GUI), while some network simulators require input scripts or scenario description (network parameters: node placement, connectivity, link failures, etc.). In particular, in the NS-2 simulator a split-level programming model is provided in which packet processing is done in the C++ system language while simulation setup is done in a scripting language (i.e., Tcl/Tk, oTcl). A common output of simulations is the trace files.

Simulators typically come with support for the most popular protocols in use today, such as IPv4, IPv6, UDP, and TCP. Some of them, as the case of [[100]; [102]; [55]; [68]; [72]], bring the support for simulating different kind of WSONs (like wireless sensor networks, vehicular networks, MANET, etc.). On the other hand, simulators like [[124]; [42]; [125]; [126]; [147]; [136]] were specifically designed for the wireless sensor networks' simulation.

Some works in the literature bring an interesting discussion about limitation and drawbacks of some simulators [[23]; [79]; [6]; [74]]. A survey on simulators was also performed by D. Curren in [[30]]. This survey presents characteristics of some main network simulators, discuss their advantages and disadvantages, and present their differences.

In spite of the drawbacks described in Section 6, simulation is still considered the most widely used methodology for evaluating network protocols. This is mainly due to its scalability and reproducibility characteristics, besides allowing protocol designers to obtain some further observations that cannot be captured by analytical models. In this way, some efforts have been performed in order to avoid pitfalls in simulation studies [[79]; [10]; [9]]. This includes for example, the validation of models and protocols implemented in simulators before their use. This will ensure that they have been coded correctly and operates in accordance with the model/protocol specifications.

4.2 Emulation

An emulator is different from a simulator in the way it runs actual application code. More specifically, it refers to the ability of a computer program or hardware environment to closely reproduce the features and behaviour of a real world devices. In this way, emulation focuses on recreating the original device environment and can be time-consuming and difficult, but valuable because of its ability to maintain a closer connection to the authenticity of the real device.

In general, in the network research domain, emulators use a simulation program in conjunction with an emulated hardware in order to observe end-to-end performance of the emulated device. An emulator may thus, trick a running software into believing

that an emulated device is really a real device. As an example, software programs can emulate microprocessors and sensor devices. This focus on exact reproduction of external behavior contrasts with simulators, which use an abstract model of the system.

A workflow describing the steps required in a protocol emulation process is similar to the one shown in Figure 3 for simulation. The differences are basically at the level of detail required at the implementation step and the kind of generated results. Simulation tools, for example, results in statistically evaluated protocols and algorithms, while emulation tools in tested implementations.

Some examples of wireless sensor network emulators are: TOSSIM [[136]], EmStar [[42]], and WSim [[146]]. Although being described as a discrete-event simulator for TinyOS applications on MICA Motes, TOSSIM is more a TinyOS emulator than a general WSON simulator. This is due to the fact that programs developed in TOSSIM can be directly targeted to Motes without modification, facilitating then the source-level application and OS debugging. EmStar is a Linux-based software framework for developing and deploying wireless sensor networks, and can be used to develop software for Mica2 motes and iPAQ-based microservers. Like WSim and TOSSIM, EmStar uses the half-simulation/half-emulation approach. The WSim simulates the hardware based on the MSP430 micro-controller (MCU) series from Texas Instruments. One of the main WSim feature is its interface with the WSNet simulator to perform the simulation of a complete sensor network. More examples of emulators are the MNE [[90]], NetKit [[99]; [111]], and the NEMAN [[115]]. Other emulators are listed in [[117]].

The real advantage of emulation is the fact it allows much higher flexibility in carrying out network tests. The software piece of an emulator allows every aspect of the network be influenced and monitored like it could be in a real network, which ensures very high accuracy.

A taxonomy of a number of emulators and a discussion about the differences of each one is provided in [[117]]. Different approaches to model mobility in emulation and real world testbeds, and an overview of different evaluation techniques are provided in [[78]; [74]].

4.3 Testbeds

In addition to theory, simulators and emulators, there is a strong need for large scale testbeds where real life experimental conditions hold.

In particular, the importance of testbed-based evaluation of network protocols/applications is gaining wider recognition in the networking research community, especially in the wireless and mobility areas. This is due to the fact that testbedding real systems face problems that do not occur in simulation. More specifically, building real systems forces you to handle cases where theoretical models are incorrect, or do not capture the right details. Thus, realistic evaluation of technologies and their mutual interactions play a major role in identifying the key performance bottlenecks.

On the other hand, real testbeds also lacks of repeatability and tight control as well as scalability, mainly caused by high costs for hardware, software, and manpower. To overcome these drawbacks, some research-oriented real network testbeds that support the development, debugging, and the evaluation of new network services were deployed: Emulab [[43]], APE [[8]], ORBIT [[103]], RON [[118]], PlanetLab [[113]], PlanetLab Europe [[112]], OneLab [[101]], and GENI [[51]].

Emulab is an interesting example of a real network testbed with public facilities. It is an experimental integrated platform that provides access to a wide range of real experimental environments like: live-Internet (interaction with RON and PlanetLab),

mobile wireless networks (6 robots are equipped with sensor capabilities), sensor networks (25 motes are available), 802.11 Wireless, etc. In addition to network testbeds, Emulab also allows the execution of emulated experiments, as well as, simulations. In particular, by using the NS-2's emulation facilities, Emulab allows the interaction of simulated networks with real networks. Thus, Emulab unifies different environments under a common user interface, integrating them into a common platform.

The original Emulab was primarily installed at the University of Utah, which is also home of most Emulab software development. Emulab, however, is now present in more than two dozen sites around the world [[104]]. Some Emulab testbeds are: the Deterlab (cyber-DEfense Technology Experimental Research laboratory) testbed [[36]] at USC and at UC Berkeley; the TIDIA/Kyatera Emulab testbed [[135]] in Brazil; the TWISC (Taiwan Information Security Center) testbed [[138]] in Taiwan, etc.

Started as a research project at the Uppsala University and partly funded by Ericsson, APE is the only existing testbed for large mobile ad hoc networks testbed [[8]]. APE contains an encapsulated execution environment, or more specifically, a small Linux distribution and tools for post testrun data analysis. It aims at making the process of performing complex real-world tests as easy as possible. It focuses on smooth deployment, high ability of customization and ability to easily run several routing protocol implementations for comparisons.

ORBIT [[103]] is a radio grid testbed for evaluation of next-generation wireless network protocols. ORBIT consists of a 400-node indoor radio layer emulator, with 64 static nodes in a grid layout equipped with wireless network interfaces, and a 50-node outdoor, full-scale network. Funded by the NSF, ORBIT is a collaborative effort between several university research groups in the New York and New Jersey region and industrial partners like Lucent Bell Labs, IBM Research, and Thomson.

The MIT Resilient Overlay Network (RON) [[118]] is a large platform funded by DARPA. Consisting in an application-layer overlay on top of the existing Internet routing substrate and having 17 sites located around the Internet, RON allows research in Internet-based distributed systems (i.e., resilient routing, peer-to-peer systems, distributed application development, etc.). Related testbeds are: X-Bone project [[148]] which provides a toolkit for rapid deployment of overlay network for things like IPv6; and the 6bone testbed [[1]], an IPv6 testbed to assist in the evolution and deployment of the next generation Internet network layer IP protocol.

The development of the PlanetLab [[113]] platform had as key motivation to evolve an improved Internet architecture by implementing new protocols as an Internet overlay. More specifically, PlanetLab is a network of computers located at sites around the world, forming a testbed for creating and deploying planetary-scale services. In this way, PlanetLab serves as a testbed for overlay networks and responds to the interest of the research community in experimenting large-scale applications. New large-scale services can then be tested and validated in an environment that is intended to replicate the environment of the Internet but does not disrupt the performance of the Internet.

A PlanetLab independent slice and management authorities spanning Europe is emerging, the PlanetLab Europe [[112]]. Its control centre is in Paris and it is also federated with the worldwide PlanetLab control centre in Princeton. This guarantees the access of the entire worldwild PlanetLab platform to joint members of PlanetLab Europe.

The PlanetLab Europe is in fact supported by the OneLab project, financed by the European Commission [[101]]. OneLab is an open networking laboratory that has as goal to extend the PlanetLab infrastructure by enabling deployment of PlanetLab

nodes in new wireless environments. In addition, OneLab also aims the improvement of monitoring capabilities of the PlanetLab, taking into account both networking and system performance issues.

Finally, the Global Environment Network Innovations (GENI) project [[51]] consists in having a bold new research platform, supported by the American National Science Foundation (NSF). GENI aims to help the construction of a 21st Century Internet, a new Internet fundamentally better than the Internet of today. The goal of GENI is to enhance experimental research in networking and distributed systems, and to accelerate the transition of this research into products and services that will improve economic competitiveness. GENI platform will consist of a collection of physical networking components, including a dynamic optical plane, forwarders, storage, processor clusters, and wireless regions.

Emulab [[43]], APE [[8]], and ORBIT [[103]] are some examples of testbeds allowing experiments in WSONs, while RON [[118]], PlanetLab [[113]], PlanetLab Europe [[112]], OneLab [[101]], and GENI [[51]] focus on Internet-based experiments. In particular, authors in [[78]; [74]] provide interesting surveys on real-world experiments and testbeds of Ad Hoc networks. The referred documents report key attributes of some testbeds and provide a testbed classification to aid researchers in selecting the appropriate candidate tools for their experiments.

4.4 Prototyping

Prototyping is the process of building a model of a system. This model is then used to test the designed aspects or to illustrate ideas of the system. In this way, a prototype represents a useful way to refine designed ideas as a preparation for the final system deployment.

Recently, the new concept of *rapid prototyping* has emerged as a powerful tool to evaluate protocol performance. Authors in [[4]] introduce the Prawn (PRototyping Architecture for Wireless Networks) software environment. Prawn allows rapid prototyping and focuses on obtaining, with little effort, an instantiation of the system that may not be optimized but is fully functional and complete. The goal is thus to make prototyping becoming attractive for designers by making systems design and evaluation easy, quick, and effortless as possible. Notice that prototyping represents a different but complementary process of protocol design compared to real testbeds.

Prawn provides a set of basic building blocks (like neighbor discovery, link quality assessment, message transmission, etc.) that implement common functionalities required by wireless protocols. By using a language independent API, the designer can then use the defined high-level primitives to send and receive data, and retrieve information from the network, without caring about sockets, communication setup, addresses, etc.

Contrarily to real testbeds, rapid prototyping addresses very early stages of the design process. It is presented as a complementary approach to simulation, emulation, or real testbeds. In opposite to the previous non-formal approaches, the main goal of rapid prototyping is to facilitate the protocol design process.

5 Formal approaches

By surveying the literature, it appears that the protocols are most of the time designed using rather non-formal approaches. In an initial phase, non-formal approaches ba-

sically consist in a textual description of services and data structure provided by the protocol. This design process is then iteratively carried on through incremental phases. During these phases, the designers informally refine the protocol by appending details and checking eventual errors. Finally, once the designers are satisfied of the service requirements and they are convinced that it is no more error-prone, the design process is terminated and the protocol is provided. Nevertheless, as this design process is based on informal refinements, it is impossible to guarantee that all requirements are globally respected. By this manner and for many protocol standards (e.g. ITU, IETF, etc.), these designed specifications contain ambiguities and omissions. Even if a programmer implementing that service will attempt to reasonably resolve the ambiguities, it will in all likelihood lead to an incorrect and inconsistent implementation. That is the reason why many efforts conducted by the Formal Description Techniques (FDT) community as well the mathematical analysts community are still devoted to formally model and validate the protocols.

Due to the flexible nature of wireless services and the high size of their user community, errors not detected through the incremental design process above mentioned, could have mighty impacts especially if they appear once deployed. In addition, although analytical methods for modeling and analyzing complex systems are known from the 60's [[109]; [93]] and though formal approaches have been used to validate wired systems for many years now [[64]], wireless self-organizing networks raise many novel constraints that open current new issues in the design, the analysis, and the formal validation of their protocols (see Section 3). Nevertheless, some formal methods and tools may be successfully adapted to be used in the wireless context. Contrarily to non-formal approaches, the main objective of formal approaches is to provide a non ambiguous model of the protocol from which quantitative or qualitative properties may be extracted and studied.

The main goal being then (1) to evaluate the performance of a system (by the mathematical analysis community), (2) to match the formally specified and extracted properties with the protocol requirements or directly with the implementation by reasoning on the system using mathematical relations. Formal approaches are thus able to guarantee that the protocol is efficient, free of errors and that the implementation satisfies the requirements by means of functional and non-functional properties. First the mathematical analysis community provides diverse techniques to model complex systems applying mathematical aspects. The main goal is to evaluate the performance of the system. Secondly, two main technique sets define the formal validation approaches that are commonly applied to network protocols: *verification* and *testing*. The goal of formal verification is to improve on reliability by arguing on mathematical logics and thereby to check that the formal system model fully complies with a given set of requirements. Regarding the formal testing, test sequences are generated from the correct and verified models in order to be checked on the real implementations.

These formal communities provide non ambiguous and correct formal models as well as implementations conformed to informal requirements. They cover the broad cycle of the protocol development and drastically enhance its reliability and efficiency. These techniques are however quite distinct and hence constitute three different communities. Their research works devoted on WSON are depicted in the following.

5.1 System modeling

The importance of modeling computer and communication systems has been increasing during the last years. It has been considered an important step for the understanding

and the prediction of systems' behavior. In particular, system modeling is the process of generating: (1) abstract methods, (2) conceptual methods, which use metalanguage to express formal interpretations, (3) graphical methods, which is the study of graphs that use mathematical structures to model pairwise relations between objects from a certain collection, and/or (4) mathematical methods, which use mathematical language to describe a system.

Models are thus, used to answer questions related to the efficiency and trustworthiness, offering means of comprehending an otherwise incomprehensible problem. It helps to visualize the problem, to break it down into discrete, manageable units. Although being a simplified abstract view of a studied complex system, a model has the required features to accurately capture the behavior of the system.

Performance and dependability evaluation is a fundamental process in system design and validation [[137]]. Sophisticated and always more powerful modeling techniques are regularly adopted in order to provide a quantitative and qualitative analysis of systems under examination. The performance of a system can be thus, improved via modeling it using mathematical methods. Mathematical models can be classified as : linear vs. nonlinear, deterministic vs. probabilistic (stochastic), static vs. dynamic, or lumped vs. distributed parameters.

Among the mathematical models, stochastic models have been found useful in the design and analysis of advanced computer and communication systems. "Stochastic" means being or having a random variable. A stochastic model is a tool for estimating probability distributions of potential outcomes by allowing for random variation [4] in one or more inputs over time. This means that even if the initial inputs are known, there are many possibilities the outcomes might go to, but some paths are more probable and others less. Distributions of potential outcomes are derived from a large number of simulations (stochastic projections) which reflect the random variation in the input(s). This large number of simulations are usually gotten by stochastic techniques called Monte Carlo methods [[37]; [120]; [81]; [142]]. A Monte Carlo method is a technique for iteratively evaluating a deterministic model using random numbers and probability to solve problems. It is just one of many methods for analyzing uncertainty propagation, where the goal is to determine how random variation, lack of knowledge, or error affects the sensitivity, performance, or reliability of the system that is being modeled.

Two special classes of stochastic models are: the Petri nets and the Markov chains. Developed in the early 1960s by C.A. Petri in his PhD. dissertation [[109]], Petri Nets are useful for modeling concurrent, distributed, asynchronous behavior in a system [[107]; [108]]. Also known as a place/transition net or P/T net, a Petri net is a directed bipartite graph, in which the nodes represent transitions (i.e. discrete events that may occur), places (i.e. conditions), and directed arcs. Arcs run between places and transitions, never between places or between transitions. The places from which an arc runs to a transition are called the input places of the transition; the places to which arcs run from a transition are called the output places of the transition.

The Markov models are some of the most powerful tools available for analyzing complex systems and are derived from the Markov property [[93]]. The Markov property states that "given the current state of the system, the future evolution of the system is independent of its history". In other words, a Markov chain is a sequence of random variables $x_1, x_2, x_3, \ldots$ with the Markov property, namely that, given the present state, the future and past states are independent.

[4] The random variation is usually based on fluctuations observed in historical data for a selected period using standard time-series techniques.

Markov chains are often described by a directed graph, where the edges are labeled by the probabilities of going from one state of a system to the other states. In discrete-time Markov chains, the changes of a state of a system are only performed in discrete time. On the other hand, in continuous-time Markov chains, changes can happen in any random exponential distributed time. A property allowing the description of a continuous-time Markov process in an equivalent discrete version can be also found in the literature, allowing the simple and efficient definition of interest measures [[35]; [18]].

Stochastic fluid models have been also largely used in the literature for analyzing different computer and communication systems [[31]; [77]]. Stochastic process algebra are also interesting mechanisms for modeling distributed systems. The Performance Evaluation Process Algebra (PEPA) [[105]] of Jane Hillston, is an example of stochastic process algebra used for modeling systems composed of concurrently active components which co-operate and share work.

In particular, when the events rate of a system vary in many orders of magnitude, the use of fluid models can result in considerable reduction of the computational cost when compared with the models where all the events are explicitly represented. In this context, an important area of application is the queueing model, used to mathematically analyze the queueing behavior of a system (e.g. the network analysis of a transmission link [[7]; [40]]). Queueing models allow a number of useful steady state performance measures to be determined, like: the average number in the queue, or the system; the statistical distribution of those numbers or times; the probability the queue is full, or empty; and the probability of finding the system in a particular state. Markov chains can also be used to model various processes in queueing model [[18]; [77]].

The main problems in modeling networks are that models are usually too large to be handled by a computer system, and, due to model complexity, model development is very time consuming. As a solution, works in the literature present methods for complexity reduction, thus reducing the development time considerably [[32]].

Regarding WSONs domain, some works in the literature use analytical models for modeling medium access protocols in wireless networks. The models are thus used to compute throughput of flows in arbitrary network topologies or specific topologies [[16]; [24]; [116]; [19]; [50]; [49]], to analyze delay in single-hop 802.11 networks [[134]], to study the capacity in mobile ad hoc networks and DTNs [[47]; [50]; [48]].

5.2 Verification techniques

There exists very few research works on the validation of wireless systems and more specifically on WSON. The reason is twofold. First, many are the ones who believe that the inherent constraints of such systems make the verification process very time consuming and that an important threshold has to be crossed before becoming proficient. Second, it was shown that there are inherent limitations of the methods for such wireless protocols [[145]]. Nevertheless, we cite some of the studies providing techniques to formally verify protocols in wireless self-organizing networks.

In order to formally validate a protocol, it has to be initially specified in an unambiguous and structured manner. Two main approaches are advocated for the verification.

- The first one called “code verification”, also known as static analysis, means that the protocol is implemented with a regular programming language such as Java and the code is verified thereafter. It becomes common that model checkers

work directly on the source code of software implementations rather than on model provided by a user or a standard. We can cite the model checkers BLAST and SLAM [[39]; [59]], which work on C source code. For Java programs, we can mention Bandera [[29]] and Java PathFinder [[60]], and Verisoft [[57]] for C/C++ programs. In such techniques that consist in constructing a model from the source code, the abstracted model can loose some information and the code could still contain ambiguities and flaw. From our knowledge, even if static analysis is widely used in other area, it has been used only once onto WSON protocols in [[44]]. The authors directly analyzed the code for finding errors in different AODV implementations [[106]]. This method has been utilized especially for bug assessment though it allows identifying a loop error in some of the implementations.

- The second approach consists on describing the protocol using formal description languages. These may be subsets of logics (first order predicate logic for instance) or subsets of automata (or transition systems) representations.

Commonly, the requirements on the system that have to be verified are expressed using a temporal logic such as CTL (Computation Tree Logic) [[27]], LTL (Linear Temporal Logic) [[114]] or TCTL (Timed Computation Tree Logic) [[5]]. These properties may be categorized as either liveness or safety [[76]]. Although safety properties can be noticeable in a finite running time of the system, liveness properties require sometimes infinite system runs (i.e. properties corresponding to *"something good will eventually happen"*).

From these formal models and the expressed properties of the requirements, two main approaches are applied. The first one is an algorithmic verification method commonly named *model checking* [[28]] knowing a broad success in the industry especially due its interesting results (possibility to easily find the flaws and help to their correction) as well its wide toolset. The second method namely *deductive verification* [[92]] consists on formally proving that the property may be drawn from a given set of premises. While the main advantage of this latter approach is to prove properties on infinite state systems for which theorem provers are processed, the advantages of model checking is to obtain a verdict in an automatic way and to quickly locate the flaws.

Figure 4 shows a summary of the verification techniques and their order of use.

5.2.1 Model checking approaches

Model checking approaches have been being more commonly used in wireless areas than deductive verification approaches. Some interesting applications to WSON protocols are depicted in the following.

Renesse et al. [[34]] have utilized the model checker SPIN [[65]] to the MANET routing protocol WARP. By specifying their protocol using the high level language PROMELA (PROcess MEta LAnguage) and their properties in LTL, the authors have covered 98% of the state space. However, the experiments have been performed for a rather small network containing indeed only five nodes.

A Bluetooth location system was also verified using SPIN and PROMELA in which the network topology consisted in 18 wireless nodes with various scatternet layouts and properties considering four basic properties [[46]]. The authors illustrated an ill-defined protocol as well as incompatible properties on some of the configurations. By

Figure 4: Verification techniques.

that way, even if very few properties were studied, it leads to guess that topologies, mobility, interference, etc. could be real issues when defining such properties in WSON.

A formal approach to verify a model health service platform has also been performed [[71]]. The main contribution is to integrate the OMG MDA (Model Driven Architecture) concepts to the validation process. Although the results are somehow not numerous with a limited testing coverage, it has the interest to raise challenges in verification when using models processing by the MDA.

The model checker UPPAAL [[84]] was applied in [[25]] in order to study the timing properties expressed in LTL of the MANET routing protocols AODV. Interesting results and possible enhancements of the protocol were provided. However, the experimented topology was linear and a restricted number of nodes were chosen.

The UPPAAL tool plus TCTL expressions were applied to verify the quality of service on a MAC protocol for wireless sensor networks [[143]]. The monitored topology considered in this paper is linear and the main properties verified on the protocol are real-time constraints especially for automotive domains. Despite some important given assumptions (particularly on the topology), an exhaustive exploration of all paths of the system execution graph was conducted in which interesting scenarios have been executed. The results especially on the timing constraints verification in such networks are very promising even if issues are still remaining. Complementary results for their verification process have been obtained in [[56]] where UPPAAL was evaluated.

In [[80]], model-checking is used for the verification of the 802.11 standard. The authors have focused on assessing the performance of the basic access (a distributed coordination function) and two-way handshake achieved through the verification of properties expressed in PCTL temporal logic [[41]]. For that purpose the authors have constructed a probabilistic model (a Markov Decision Process) referring to a specific and fixed topology consisting of two source and two destination nodes. Nevertheless, due to the complexity of the study and the size of the probabilistic timed automata, the scalability is uncertain.

The authors in [[11]] described the use of probabilistic model checking for the comparative analysis of an IEEE 802.11 set of protocols for MANETs with S-MAC a protocol designed to reduce energy consumption suitable for wireless sensor networks on a 3-hops topology. The formal specification is a Markovian model which is analyzed using a probabilistic model checker like PRISM [[121]]. The model verification has been achieved through a combination of specific probabilistic reachability PCTL properties and rewards properties. However, the main drawback (common to many other works) is still the scalability, the topology applied in that paper being indeed relatively limited (four nodes, one source, one sink).

5.2.2 Deductive verification approaches

Deductive verification techniques are less frequently applied. The main reason that may be put forward, is that the theorem prover tools are not self-sufficient and because of the complexity of constructs or functions designed, experts are most of the time required to manipulate such tools. Nevertheless, we may cite few research works for WSON in this area.

Bhargavan et al. [[14]] applied the HOL [[63]] theorem prover in their deduction of route validity and freedom from routing loops in AODV. They utilized three conditions on next node pointers, hop counters and sequence numbers to form a path (namely the *invariant* in deductive verification) on pairs of nodes (on the path from source to destination). Three properties were verified using SPIN after which HOL has allowed

to prove that the three properties led to the path invariant theorem. But an important contribution is to illustrate that WSON protocols are quite difficult to be specified. Indeed, despite the interesting results provided by the authors, they noticed themselves that they fail to prove properties for AODV (while it was a success for RIP).

Another work for the MANETs has proved the absence of routing loops in a simplified version of AODV [[33]]. The strategy is quite similar to the previous one, but more automated. They have used predicate abstraction and could raise most of the quantified predicates automatically by analyzing spurious counter-example traces. The method has successfully discovered all required predicates for the considered version of AODV but in a rather limited network that is with 3 nodes and without any mobility.

Following the previous example, a tool called UCLID PA [[122]] was performed to prove the freedom of loops in AODV once again [[82]; [83]]. Whilst the previous verification process was done using quantified predicates, the authors used indexed predicate abstraction providing therefore an enhanced axiomatic system but applied on the same very small and limited network.

Borujerdi et al. [[20]] have verified properties extracted from the MobiCast protocol [[132]] integrated in a mobile network with micro-cells. The protocol has been formally specified in Prolog language [[131]] and expressed properties proved by applying an SLD resolution [[129]]. Several inconsistencies were detected in the first version of the protocol and resolved with minor variation. And despite the fact that no real mobility is applied to their experiments, the authors demonstrated that formal approaches are required all along the development phase of a protocol.

5.2.3 Hybrid approaches

Although model checking and deductive methods are commonly employed separately, an attempt for merging both techniques to wireless network domain has been studied. McIver and Fehnker [[96]] defined a stepwise specification and refinement of WSON protocol characteristics using a combination of proof-based methods and model checking. Energy-efficient protocols in wireless networks have been considered especially by analyzing delays and collisions criteria in such systems. From an exhaustive search with model checking, the authors have illustrated weaknesses in the systems and thus have provided lower and upper bounds on quantitative aspects of the protocol. Besides, formal proofs have enabled to investigate optimal protocol behavior especially in terms of trading off energy requirements and performances. Nevertheless, both methods being quite different, issues regarding the obtention of realistic formal models and performance criteria for such WSON were raised.

5.3 Formal testing techniques

Formal techniques for protocol testing are performed for a long time now [[64]]. Two main mechanisms may be utilized for such purpose: *passive* and *active* testing.

Active testing is based on the execution of specific test sequences to the implementation under test. These test sequences are obtained from the formal model according to coverage criteria. These criteria can then be applied on the specification, e.g. coverage of all logical conditions, coverage of all paths. This allows setting if the specification as well as the code were covered during the testing phase.

The tests may be generated automatically or semi-automatically from test criteria, hypothesis, and test goals. The format of these sequences which is commonly used

by the testing community is TTCN3 [[2]], from which their execution are performed through points of control (execution interfaces). These points of control are installed in the context of a testing architecture, which means the way to put the testers (e.g. upper and lower testers to test a specific stack layer, the different interfaces, and the oracle in order to provide a verdict on the executed tests to the tested implementation). We can mention two families of testing: the *static* and *dynamic active testing*. The first one is based on static analysis of the source code, i.e. the implementation. The code is inspected regarding the elaborated checklist or by analyzing the control and data flow graph. Using this kind of test, we do not have to exercise the system under test with real data. On contrary, dynamic testing implies that the system under test is executed under different configurations, i.e. with different input data tests. The tests sequences to be exercised on the implementation are derived from the model described by a FDT. Afterwards, the inputs of the test sequence are given to the implementation and the output results are compared to those expected by the specification.

Passive testing consists in observing the input and output events of an implementation under test in run-time. The term “passive” means that the tests do not disturb the natural run-time of a protocol or service. This concept is sometimes also refereed to as monitoring in the literature. The record of the event observation is called an event trace. This event trace will be compared to the formal specification as a test sequence. The passive testing techniques are applied especially because the active ones require important testing architectures, whose the testers need to control the system at some specific points. This is sometimes not feasible or even undesired. Nevertheless, while test sequences in active testing may give concrete verdicts (excepted for "*inconclusive*" ones), an event trace that satisfies the model does not mean that the whole implementation satisfies the specification. On the other hand, if a trace does not satisfy, then neither does the implementation.

Passive and active testing have their own advantages and drawbacks especially when used to wireless self-organizing network protocols. Nevertheless, the results that may be obtained depend on the system under test and essentially on the testing goal, the testing type. The testing type considers the whole testing process of the protocol, which consists in different steps: unit, conformance, interoperability, integration testing, and so on. Most of these testing types are normalized. For instance the conformance testing is standardized by the ITU-T in [[3]] in which common testing architectures, interfaces or points of control and observation are mentioned and specified. Nevertheless, these standards are mainly designed for wired systems and most of the time (if not always), the new inherent constraints of WSON (such as the lack of infrastructure or non-centralized systems) are omitted from these documents. Although the tasks for testing in such an environment seem tough, studies on formal techniques for testing protocols in such wireless contexts are presented in the following.

Figure 5 shows a summary of the formal testing techniques.

[[85]] proposes a formal methodology to specify and analyze a MANET routing protocol. It is based on the Relay Node Set (RNS) concept. A RNS is a set of nodes allowing reaching all nodes in the network. According to the studied protocol, the set is built differently: the reactive protocols build the set in a regular manner, when it is required; while the proactive ones build the set during the route discovery performed at the beginning of the network lifetime, being maintained during the network lifetime. Passive testing techniques for conformance testing are applied in that paper. The framework illustrated in that paper has as main goal to analyze the implementation under test by metrics through non-functional aspects which are usually applied for a performance analysis.

Figure 5: Formal testing techniques.

In [[54]], the authors developed a formal model namely Distributed Abstract State Machines (DASM) to allow the specification of ad hoc network routing protocol. Besides, their motivation was to raise the main issues when designing such kind of protocols. Nevertheless, only conformance testing by an active mechanism has been processed. While the model allows verifying the behavior of a node in a functional way, it is non-executable and does not allow observing the interoperability of nodes.

[[151]] is another approach to formalize testing of a MANET routing protocol by applying game theory concepts. The game theory is based on the "income" calculus. "Income" means for instance the convergence when the topology is modified or the induced overhead. A strong hypothesis commonly applied in game theories consists on the complete knowledge by the "players" of the "game". It means that each node is supposed to have a complete knowledge of the network topology as well the nodes or links states. Furthermore, complete knowledge forbids the non-determinism. Therefore, some mobility models based especially on non-deterministic behaviors and commonly used in WSON analysis can not be modeled. In many wireless self-organizing networks and because of their characteristics, the complete knowledge assumption may not hold, which invalidates the test sequences execution.

In [[87]], the authors presented a new methodology for the conformance testing of a MANET routing protocol, namely Dynamic Source Routing (DSR). Active mechanisms were applied combined with a nodes self-similarity approach allowing avoiding the so called issue "state space explosion" and reducing the needed formal model to a dynamic topology. The approach was applied on the DSR-UU implementation [[38]]. Interesting ideas for interoperability testing contexts were discussed.

Merouane et al. [[97]] proposed an interoperability testing approach based on an EFSM (Extended Finite State Machine) based language. A testing architecture was provided and test sequences automatically generated from an SDL specification in order to test an implementation of DSR-UU. This work reveals the real complexity of studying interoperability aspects in the WSON. Some of the systems are so dynamic that the observed or controlled linked (through the interfaces of nodes) become a hazardous task. They therefore employed an emulation mechanism coupling NS-2 simulator and the implementation under test integrated in two real nodes.

5.4 Hybrid techniques

Although these verification and testing techniques quite differ regarding their approaches and their own objectives, they are also complementary. That is the reason why interesting works have been occurred by presenting the advantages of mixing verification and testing approaches. Therefore, in [[89]], the authors have defined a stepwise approach that uses two languages to specify the OLSR protocol [[133]]. They have applied Promela associated to the powerful tool SPIN to perform verification and SDL associated to the ObjectGeode tool [[140]] to generate test sequences. Since both languages are different, the design approaches have not been the same and a compromise has also been required through the specification choices. Finally, the work illustrates essential issues while covering the protocol lifecycle. Indeed, the authors reveal that the languages do not have the same expressiveness power and that some aspects which can be shown in a form with one language could be impossible with another one. This is therefore why these validation activities are most of the time completely distinct.

6 Objectives, limitations, and drawbacks

In the previous sections, we have presented non-formal and formal approaches commonly used by the network and the performance analysis communities to address non-functional properties. In the section devoted to formal approaches, some relevant works coming from verification and formal methods communities were presented, which have been carried on to formally verify functional properties regarding protocols' behaviors.

This section reminds the main objectives of techniques used in formal and non-formal approaches. The limitations and drawbacks of such approaches are then enumerated and discussed with regard to WSONs domain.

6.1 Regarding non-formal approaches

Simulation as explained previously is a technique widely used to evaluate performance and seems to give good results for a large number of protocols [[73]]. It models the influencing factors and algorithms and investigates these aspects in an artificial software environment with a high degree of abstraction. This technique appears at the same level of the formal verification technique of the protocol development cycle. Simulation is also widely used because the manipulated concepts are easier to model the system than analytic analysis, based on mathematical equations. The main advantages of this technique are that it permits to observe and analyze the protocol performances and to verify expected properties (e.g. network lifetime extension, data delivery). Moreover, it is much cheaper than other techniques as it allows reproducibility and repeatability with a low effort just by changing some models parameters.

Emulation provides a best compromise between cost and precision. Emulation is a mix of software and hardware combined to reflect the behavior of a network. Some of the components are real components and others are simulated. The main objective is thus, to test protocols in a more realistic environment, as a way to better forecast and analyze protocol behavior, to later real world implementation. This technique inherits the drawbacks of simulation that will be detailed below.

Real world testing consists on performing tests in a real testbed, which is an implementation of the system very similar to the final one. The dimensioning and the configuration of real testbed experiments are in general, hard and costly processes.

Since potential problems are only detected at the end of the lifecycle of the system, if a built system is not the expected one, the implementation phase should be restarted again from the beginning.

To summarize the techniques coming from the network research and performance engineering area, we can clearly establish that the common objective is to evaluate or to verify properties that are commonly named as non-functional properties, such as: servers loads, response times, rates of loss, data delivery reliability, etc.

Limitations and drawbacks

1. **Simulation techniques**

   - The main drawback that can be formulated against simulation technique [[74]] is based on the assumptions that are made on the underlying protocol layers implemented on the simulators. More specifically, they represent potential error-prone points. In fact, models used by simulators have usually not been previously validated, which may add errors to the simulated environment. Hence, this can impact the protocol performance evaluation and finally, results in false analysis.
   - We can also mention that simulation as explained in [[144]] does not capture the internal behavior of the system. Indeed, some aspects of the system that are relevant to performance evaluation may not be directly observable, constituting then "black boxes". Hence, performance evaluation related to unit measurement can be strangled by the "black box" effect of the system. To overcome the lack of observation, the designers can decide to use a lower level of abstraction to model its system. By reducing the level of abstraction, they can be too much close to the real implementation.
   - The main concern with simulations is, however, the level of confidence we can have in them. Some works in the literature show that divergences in obtained results exist between different simulators. The main conclusions are: "*omitting detail or oversimplifying the simulation model can lead to ambiguous or erroneous outcomes*", "*simulation assumptions always affect research outcomes*", and finally, "*insufficient statistical analysis of independent simulation runs and improper data collection techniques can produce ambiguous or inaccurate conclusions*" [[23]; [79]; [6]].
   - Although non-formal approaches are highly suitable for achieving performance measures in large-scale networks, they cannot easily prevent protocol design errors. Indeed, there are properties of protocols that do not relate to performance.
   - Packet loss is commonly observable and controllable by using simulation tools [[86]] . Nevertheless, the implementation of IEEE 802.11 in NS-2 is rather unrealistic in a way that several properties (e.g. fluctuating links or different transmission rate) can not well be represented as it depends on the implementation of the radio propagation.
   - From the physical layer viewpoints, the reality (i.e. mobility and environment models) is really tough to describe. Furthermore, wireless propagation models in simulators are often too simple and general, being dependent on the details or granularity required and integrated into the simulators [[23]]. Therefore, qualitative and quantitative divergences between

the results obtained from different simulators might be observed even by applying the same scenarios and parameters.

2. **Emulation techniques**

   - The main concern with emulations is, however, complexity. Emulators are complex pieces of software, but valuable because of their ability to maintain a closer connection to the authenticity of the hardware device. Writing an emulator is also a difficult process which requires obtaining appropriate system information and then figuring out how to emulate the system hardware with software code. In addition, the computational resources needed to run an emulated device are typically higher than those available in the device itself. Consequently, the performance of an emulated device is, in general, lower than that of the real one. This may pose limits on the scalability of the size of the emulated network [[117]].
   - Emulators have to face timing constraints [[74]] that need to be thoroughly stated. Thus, due to these real time constraints, the simulation parts can be rapidly overstepped which can lead to scalability problems.
   - From the physical layer viewpoints, the reality (i.e. mobility and environment models) is really tough to describe. Furthermore, wireless propagation models in simulators are often too simple and general, being dependent on the details or granularity required and integrated into the simulators [[23]]. Therefore, qualitative and quantitative divergences between the results obtained from different simulators might be observed even by applying the same scenarios and parameters.

3. **Real world testing techniques**

   - While real testing experiments are ideal to tackle errors invisible in simulation, experiments become rather limited as the number of nodes increases [[74]]. Furthermore, these experiments have to be carefully executed and orchestrated.

6.2 Regarding formal approaches

As previously described, there exists various techniques to design a protocol by means of formal methods. This section rapidly sketches the different techniques that are available and emphasizes their limitations, and how they step in the development lifecycle.

In the area of formal methods, verification and testing techniques have been widely used for wired protocols and are also used in WSONs. Section 5 has identified several works on these two main domains that have tried to transpose well known techniques to WSON protocols. Nevertheless, this is not a straightforward rearrangement due to the new challenges introduced by WSONs (see Section 3), which have underlined new limitations or exacerbated the existing ones.

Verification consists first in providing a model of the system. At this stage, the model will be used to automatically verify some properties of the protocol on the model and not its performance. By this technique, we can verify the correctness of the protocol regarding the requirements of the desirable behaviors of the protocol. The model issued from this step can be used to generate the tests. The active testing generation phase has to deal with a model that has been firstly checked, i.e. the model is conforming to its requirements. Furthermore, the model to generate the tests can be constructed

only for testing purposes and not coming from the verification. The languages used for test generation and for verification goals are often not the same. Nevertheless, the model from which the tests will be generated requires being verified regarding the adequacy to requirements and also regarding some inherent properties in the protocol, such as loops free, or deadlocks free.

In comparison to verification by model checking, deductive verification can handle infinite state systems with very rich data structures. The main drawback of this approach, which can explain that its use is not widespread, relies on the design by the user of constructs, such as invariants and ranking functions. In other words, it requires very accurate users' skills to guide the theorem prover tool as the process is not fully automatic.

The techniques, that operate to generate test cases and to verify some properties expressed by logical formula, are based on exploration of the state space of the model. Both techniques have to face often the well known problem of state space explosion. The test generation methods produce test cases that are called abstract tests. To be executed on the real implementation, these tests have to be expressed in a concrete way, notably in the programming language used by the implementation, in order to be able to interact with this one. During the execution step, the concrete tests are executed by feeding the implementation with the input of the test case and the answer of the implementation is compared to the expected one. This way to communicate with the system is not always feasible and to overcome this drawback, we can mention passive testing techniques. Indeed, passive testing needs only to retrieve traces of the real implementation by means of *sniffers* and seeks on these traces the expected behaviors of the implementation.

Whilst non-formal approaches are essentially dedicated to non-functional properties, the traditional concerns of formal techniques are to establish the correctness of system regarding the expected behaviors. We identify these properties as functional, which, if considering communicating systems, are related to the system behavior in terms of interactions.

Limitations and drawbacks

- Modeling is a time consuming activity that requires a tremendous effort, but that can not be avoided at the model-checking and model-based testing phase. Although constituting an essential phase to establish the correctness of the protocol, modeling is a very complex activity especially regarding the level of abstraction to be considered.

- In order to address the new challenges raised by WSON, formal methods have to face the well known problem of state space explosion. The explosion can occur when the state space storage grows exponentially with the size of the model. To overcome this problem, many techniques have been proposed such as symbolic representation, partial order reduction, compositional reasoning, abstraction or symmetry [[65]]. Nevertheless, we are still faced with this problem. This explosion is also responsible of non exhaustivity of the tests.

- Existing modeling languages have to deal with new features of WSONs. Whatever the techniques used, the language needs to be expressive enough to capture all the WSON features. Moreover, a tool is needed in order to automate verification and test generation on the model. It also has to be able to handle the logic used to express properties to be checked and also to cope with the language used

to express test objectives. This aspect is very important in particular for active testing techniques.

- Languages used by verification techniques are not able to handle the broadcasting mode of wireless communication. Many languages used by verification do not allow performing such type of communication. The way to perform such broadcasting could be to specify it by unicast link with a matrix that contains the identifier of all the nodes around the network. This is still an open issue [[145]].
- Many models consider fading links with distance between receiving or sending nodes. Nevertheless, message losses due to collisions in a communication between in-range nodes are most of the time obscured by the model. In addition, the collision model is commonly required to challenge particular fault-tolerance mechanisms [[143]].
- Whilst models for formal verification may easily describe timing aspects designed for WSON protocols, languages dedicated to testing approaches are not always (and most of the time) adapted. Constraints such as the execution on a real system might suppose that the nodes are synchronized which is a WSON challenge [[86]].
- The mobility of nodes in WSONs generates topology changes. For the modeling and verification, we have to consider such change of connectivity. This features related to the kinetic of nodes can increase the state space explosion and there is no straight way to address it. Nevertheless, in [[88]] the authors explain a way to cope with this drawback.
- Concerning passive testing approach, we have to face a lot of inconclusive verdicts when the collected traces are too short. In this case, the reasoning of a correct or faulty implementation is not possible. Moreover, the way to characterize the properties that are found on the traces are known as functional patterns. They characterize the expected behavior of the system. They have to be expressed in an efficient manner, i.e. in a non ambiguous way. This activity relies on a strong expertise of the system to be tested. Nevertheless, to overcome this difficulty some works try to assist this step by providing automatic mechanisms to verify the soundness of the patterns on a formal model [[12]]. Another drawback of such techniques relies on the difficulty to merge the off-line traces of a multihop wireless network. A recent work [[150]] has been done on the correlation of traces between two distant entities. For the WSONs, in our knowledge, there is no work that show how to correlate traces coming from different nodes in order to verify global properties.

7 Domains convergence

In practice, formal and non-formal domains work independently, being few cross-fertilization existent between them. Nevertheless, several attempts in the research area of wired protocols have been made to converge these two domains, usually accomplished by researchers coming from the formal area [[21]]. This paper thus, emphasizes the experiences that have been conducted to make them converge. In this way, this section reports these experiences and provides discussion about the common required effort to achieve that cross-fertilization for validation of WSON protocols.

The observations that have conducted to such research is based on practical experiences. Indeed, design experts communicate rarely with performance experts. The obstacle cannot only be explained by the difference in the concepts that they manipulate. Formal design methods and performance design do not address the same objectives. The first one is devoted to establish the correctness of problems by revealing errors. Otherwise, performance evaluation allows first avoiding system malfunctions due to over congestion of resources [[22]].

Performance evaluation for software systems allows efficiently characterizing the right configuration of the system to respect well-defined quality of service. Moreover, we can identify congestion or faulty management resources. As mentioned in [[144]] designers have often neglected performance engineering. Discovering a performance problem very late in the development lifecycle of a protocol can be costly. In [[22]], they stated that 80% of client/server systems have to be rebuild because of low performance obtained under that required. In this way, software performance models of an early design can decrease the performance failures. Indeed, the performance of a system can be strongly related to the architecture of the system. Ideally, the performance should be considered at a early stage of the design of the system.

After these statements, in the following sections, we address the attempts of convergence in several manners and at different levels. We first report on the works performed at the modeling step, i.e. on the description of what the system is expected to do. In the formal area, the model is called a formal specification. The way to take into account performances requirements at the specification level consists of annotating the model with performances requirements. Afterward, we report on testing experiences. We explain how, for correctness purposes, some connections between both communities have carried on.

7.1 Convergence at the modeling step

We can notice that the link existing between formal techniques and simulation lies on the use of a model. This relationship is a good starting point to make them converge, even if the objectives are not the same. One of the idea beyond is to try to take benefit of the performed work to establish the correctness regarding functional properties, and then, to use this correct model to evaluate performance-related issues. Otherwise, another idea is to take advantage of the work performed at the performance level, by completing the existing and preliminary formal model with the performance model.

Some works to bridge both communities have already been proposed for the integration between non-formal and formal approaches in the case of validation of wired-based protocols. By surveying the literature, a lot of works dealing with transformation of functional models into a performance oriented model can be found [[21]; [61]; [110]; [98]; [130]]. Especially, the integration consists of adding at formal model non-functional properties. With such an approach, the work performed at a modeling step for validation purpose is reused for performance evaluation. We do not need to perform twice the modeling work. As the purposes are not the same, the modeling step needs some adjustments to deal with performance constraints. It also needs to have an analysis method to solve the resultant model. Typically, the performance evaluation is performed after the functional design, when all the architectural decisions have been done.

The convergence of network research and protocol engineering research is becoming more and more necessarily especially with the new challenges that the WSONs raise. The techniques are faced to their limits. In this way, we argue the importance of taking advantage of the differences of techniques. This is also the observation done by

the authors of [[123]]. This paper is indeed, from our knowledge, the only one that applies formal models to present the different stack layers of a WSON node (in this case for the sensor networks) and their environment as well. The authors advocate to use the FDTs for all their advantages such as the reliability, reusability, etc. (see Section 2). The basic formalism is the communicating parallel interpreted automata written in ReactiveML [[91]] and the hardware functionalities may be designed in VHDL (Very high speed integrated circuits Hardware Description Language). An interesting aspect is to formalize the environment which may also be the cause of the different results obtained in simulation and real case study. For that, the Lucky tool [[69]] is proposed. Finally, and like it is introduced in [[123]], the convergence needs to occur not only at a modeling phase but it has also to be considered for validation purposes.

7.2 Convergence at validation level

Although formal and non-formal approaches have their own main testing targets as well their own characteristics and limitations, some endeavors have been performed. The main goal is to get the advantage of their differences for covering as much as possible the phases of the protocol validation in its development lifecycle. But merging or blending both methods is not effortless. First, three entities are involved in this process: the two models provided by both communities plus the implementation under test. It means that the models have to be "equivalent", even if the design languages are quite different. Moreover, the system environments must be similar which is not always simple regarding the WSON challenges and the limitations for both testing methods. It is as well required that these three entities are correct and conform to each other which is tough due the different languages, viewpoints, power expressiveness, etc. Finally, and maybe one of the main problem is that, both communities are very often distinct. Moreover, whilst it is needed to clarify some terms of vocabulary, studies on eventual common languages, models, or tools are necessary. We illustrate in the following some research attempts dealing with the mixing of both techniques.

The first step to orchestrate such bridge-building is to formalize the methodology for designing WSON protocols. Efforts have been achieved in that way by re-implementing the protocol stack layers of a simulated platform by utilizing formal refinement-based methodologies [[127]]. This work enables the testers to perform architecture exploration much quicker than using informal design techniques. Furthermore, the objective of this platform is to provide a entire product containing SW/HW. This formal design approach allows the correct partitioning between SW and HW and hence, prevents errors from being raised. Therefore, though this approach is not particularly dedicated to the validation of WSON protocols, it reveals that the design efforts for observing and/or controlling every functional and non-functional protocol aspects are necessary but, costly.

By the way, research efforts have also been provided to cope with the language transformation mechanisms or their diverse and irrelevant expressiveness power. Fehnker and al. [[45]] propose a single top-level graphical model for simulation and model checking with the ability of easily collect, observe, and analyze the results. This kind of approach is mentioned as a "bridging-language" by the authors. In fact, it is currently quite relevant regarding their results and also in taking into account the applied interference model which has been verified and experimentally validated. Nevertheless, one of the issues and not the least is that a new simulator and dedicated specification language are used. Once again it raises the difficulty for both communities to merge their efforts by making evolve their own specific tools and languages.

Another interesting research work is devoted to apply a formal testing approach merged with an emulated NS-2 platform [[87]]. Nevertheless, a specification is confronted to the huge number of topologies and as previously mentioned, the generation of test sequences may lead to a state space explosion. The authors therefore apply the concept of nodes self similarity that allows mapping a route viewed by the implementation under test (IUT) for a given message in a real network with a path of the specification containing only three nodes. Hence, since the execution of test sequences (generated from this minimized formal model) on a real implementation is rather tough in life testing, the authors advocate for applying them through an emulated platform connected to a simulated mobile ad hoc network. A MANET routing protocol was tested by this manner. Notwithstanding, still many testing inconclusive verdicts have been provided. This is especially due to: the unreliability of radio communications and because of the unexpected messages sent in the real network and not planned in the minimized model. This shows up that even if we map formal models on a informal tool (such as a simulator/emulator), aspects concerning the other running protocols (such as 802.11) have to be taken into account through the formal approaches.

7.3 Ways for converging

As mentioned before, non-functional aspects are mainly studied through performance-based tools and mathematical analysis. Nevertheless, formal approaches are more and more concerned on QoS requirements and performances [[96]; [95]; [143]; [14]]. Unfortunately, it shows up that techniques to particular performance criteria for WSON are still not tailored enough. Indeed, even if verification techniques may be applied on the models checking then the correctness of the requirements, the validation aspects are not reached especially due to non-realistic and incomplete formal models.

Based on this observation, it is interesting to reason backwards in studying works dealing with functional properties through non-formal approaches. Actually, this is met in almost all papers describing results obtained from simulators/emulators. Indeed, functional behaviors are implemented in those tools and performance results are based on these behaviors. That is the reason why the authors of Verisim [[15]] have evaluated the correctness of the functional properties implemented in such performance-based tools. Verisim is a model combining NS-2 and the trace verification component provided by the *Monitoring and Checking* system namely MAC [[75]]. The goal is to generate a NS-2 trace T and to verify if the expected properties are included in the implementation I according to a scenario S. Finally, they have shown that the AODV implementation in NS-2 was false regarding some properties as for instance the initial value for the hops number in a RREP. This work is very interesting and rather disturbing, raising several issues regarding the efficiency/reliability of the simulation/emulation concerning the correctness of the functional behaviors.

But, what does it mean exactly? Are all previous results obtained with NS-2/AODV false, and therefore should not be studied anymore? Actually not, because even if the performance results are noticeably distorted, a global understanding of the non-functional aspects are notwithstanding provided. Nevertheless, the most relevant meaning in that observation is the necessity of merging both formal and non-formal approaches to deliver the best results concerning both functional and non-functional WSON properties.

Nowadays, we may notice that whilst most of the functional properties are studied applying formal approaches, the non-functional ones are mainly tackled through non-formal techniques. And despite the increasing needs, especially with the WSONs

becoming a trend and a matter of necessity in many domains, the melting idea is rarely met. "Rarely" because we may indeed cite one very promising work [[26]]. In this paper the authors first present a technique to associate the model driven development methods and a formal description technique (SDL) to design and specify an ad hoc protocol. But the very interesting step is the use of these first results to tackle model-driven performance simulations through NS-2. Indeed, the main result is to transform a complete SDL formal specification in a NS-2 executable model. However, even if that first paper is relevant in a way for converging both worlds, many works are still to be done. Therefore, to cope with this needed blend, many targets may be pointed at. First, the used languages have to be commonly accepted by all communities. That is, it is a non sense of studying different aspects of a same protocol (or worse, the same aspects) by using different languages. The languages should evolve (or be transformed from one phase to another) being able to take into account functional and non-functional properties. In that way, these languages would be applied to check the requirements (for verification), to generate test sequences (testing), and to analyze the performance of that protocol. Thus, formal and non-formal techniques would be used harmoniously providing common and real results.

8 Conclusion

Wireless Self-Organizing Networks (WSONs) have attracted considerable attention from the network research community. Nevertheless, the success of WSON-related applications is strongly related to the well-done validation of properties of the involved network protocols. In this paper, we have investigated the existent validation approaches and provided discussion about their components and similarities. We have provided discussion about the particularities introduced by multi-hop wireless networks and how to take advantage of similarities between the validation approaches to obtain complementary techniques. In summary, our goal was (1) to provide a discussion about the *foundations* of validation techniques and the difficulties imposed by WSONs characteristics, and (2) to give hints of open research problems.

References

[1] 6bone testbed. `go6.net/ipv6-6bone/`.

[2] ETSI/ES 201 873-1. Methods for testing and specification (mts); the testing and test control notation version 3; part 1: Ttcn-3 core language, v3.2.1. Technical report, ETSI, 2007.

[3] ISO/IEC 9646-1. Information technology - open systems interconnection - conformance testing methodology and framework - part 1: General concepts. Technical report, ISO, January 1994.

[4] F. B. Abdesslem, L. Iannone, K. Obraczka M. D. de Amorim, I. Solis, and S. Fdida. A prototyping environment for wireless multihop networks. In *Proc. of AINTEC*, pages 38–43, November 2007.

[5] R. Alur and D. Dill. Automata for modeling real-time systems. In *Internation Colloquium on Automata, Languages and Programming*, volume 443, pages 321–335, 1990.

[6] T. R. Andel and A. Yasinsac. On the credibility of manet simulations. *IEEE Computer*, 39(7):48–54, October 2006.

[7] R. M. M. Le ao, E. de Souza e Silva, and M. C. Diniz. Traffic engineering using reward models. In *Proc. of the International Teletraffic Congress*, pages 1101–1112, 2001.

[8] APE. Ad hoc Protocol Evaluation testbed, `apetestbed.sourceforge.net/`.

[9] R. Bagrodia and M. Takai. Position paper on validation of network simulation models. In *DARPA/NIST Network Simulation Validation Workshop, May 1999.*, 1999.

[10] O. Balci. Verification, validation and accreditation of simulation models. In *Proc. of the 29th Conference on Winter Simulation*, pages 135–141.

[11] P. Ballarini and A. Miller. (towards) model checking medium access control for sensor networks. In *Proc. of the 2nd IEEE international symposium on leveraging applications of formal methods*, pages 256–262, Paphos, Cyprus, Oct 2006.

[12] E. Bayse, A. Cavalli, M. Núñez, and F. Zaïdi. A passive testing approach based on invariants: application to the wap. *Computer Networks*, 48:247–266, 2005.

[13] F. Benbadis, M. D. de Amorim, and S. Fdida. Elip: Embedded location information protocol. In *IFIP Networking*, June 2005.

[14] K. Bhargavan, D. Obradovic, and C. A. Gunter. Formal verification of standards for distance vector routing protocols. *Journal of the ACM*, 49(4):538–576, July 2002.

[15] Karthikeyan Bhargavan, Carl A. Gunter, Moonjoo Kim, Insup Lee, Davor Obradovic, Oleg Sokolsky, and Mahesh Viswanathan. Verisim: Formal analysis of network simulations. *IEEE Transactions on Software Engineering*, 28(2):129–145, 2002.

[16] G. Bianchi. Performance analysis of the ieee 802.11 distributed coordination-function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, 2000.

[17] B. Boehms. A spiral model of software development and enhancement. In *IEEE Computer*, volume 21, pages 61–72, May 1988.

[18] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley and Sons, 2006.

[19] R. Boorstyn, A. Kershenbaum, B. Maglaris, and V. Sahin. Throughput analysis in multihop csma packet radio networks. *IEEE Transactions on Communications*, 35(3):267–274, 1987.

[20] M. M. Borujerdi and S. M. Mirzababaei. Formal verification of a multicast protocol in mobile networks. *International Journal of Signal Processing*, 1(4):212–218, January 2004.

[21] M. Bütow, M. Mestern, C. Schapiro, and P. S. Kritzinger. Performance modelling with the formal specification language sdl. In *IFIP TC6/ 6.1 the 16th international conference on formal description techniques IX/protocol specification, testing and verification XVI on Formal description techniques IX : theory, application and tools*, pages 213–228, Kaiserslautern, Germany, 1996. Chapman & Hall, Ltd.

[22] A. Cavalli, A. Mederreg, F. Zaïdi, P. Combes, W. Monin, R. Castanet, M. MacKaya, and P. Laurenc̆ot. A Multi-Services and Multi-Protocol Validation Platform - Experimentations Results. In *The 16th IFIP International Conference on Testing of Communication Systems*, pages 17–32, Oxford, March 2004. Lectures Notes in Computer Science.

[23] D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of manet simulators. In *Proc. of the 2nd ACM international workshop on Principles of mobile computing (POMC)*, pages 38–43, October 2002.

[24] Claude Chaudet, Isabelle Guérin Lassous, Eric Thierry, and Bruno Gaujal. Study of the impact of asymmetry and carrier sense mechanism in ieee 802.11 multihops networks through a basic case. In *Proc. of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 1–7, 2004.

[25] S. Chiyangwa and M. Kwiatkowska. A timing analysis of aodv. In Springer-Verlag, editor, *Proc. 7th IFIP WG 6.1 International Conference on Formal Methods for Open Object-based Distributed Systems*, volume 3535, pages 306–321, Athens, Greece, June 2005.

[26] Dennis Christmann, Philipp Becker, Reinhard Gotzhein, and Thomas Kuhn. Model-driven development of a mac layer for ad-hoc networks with sdl. In *Proc. of 1st Workshop on "ITU System Design Languages"*, Geneva, Switzerland, Sep 2008.

[27] E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In *Proc. 1st Workshop on Logic of Programs*, volume 131, New York, USA, May 1981.

[28] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. 1999.

[29] James C. Corbett, Matthew B. Dwyer, John Hatcliff, Shawn Laubach, Corina S. Puasuareanu, Robby, and Hongjun Zheng. Bandera: extracting finite-state models from java source code. In *Proc. of the International Conference on Software Engineering*, pages 439–448, Limerick, Ireland, jun 2000. ACM.

[30] David Curren. A survey of simulation in sensor networks. Technical report, University of Binghamton, www.cs.binghamton.edu/ kang/teaching/cs580s/david.pdf, 2005.

[31] A. P. C. da Silva, R. M. M. Leäo, and Edmundo de Souza e Silva. An efficient approximate technique for solving fluid models. *SIGMETRICS Perform. Eval. Rev.*, 32(2):6–8, 2004.

[32] Ana Paula Couto da Silva. *Computational Methods for Markov Reward Models*. PhD thesis, Federal University of Rio de JaneiroBonn, 2006.

[33] S. Das and D. L. Dill. Counter-example based predicate discovery in predicate abstraction. In Springer-Verlag, editor, *Proc. 4th International Conference on Formal Methods in Computer-Aided Design*, volume 2517, pages 19–32, Portland, OR, USA, Nov 2002.

[34] R. de Renesse and A. H. Aghvami. Formal verification of ad hoc routing protocols using spin model checker. In *Proc. 12th Mediterranean Electrotechnical Conference*, Dubrovnik , Croatia, May 2004.

[35] E. de Souza e Silva and H.R. Gail. *Transient Solutions for Markov Chains*. In W. Grassmann, editor, Computational Probability, Kluwer, 2000.

[36] Deterlab. Network security testbed, `www.deterlab.net/`.

[37] Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo methods in practice*. John Wiley and Sons, 2001.

[38] E. Nordström. DSR-UU v0.1. http://core.it.uu.se/core/index.php/DSR-UU. Uppsala University.

[39] C. Eisner. Formal verification of software source code through semi-automatic modelling. *Software and System Modelling*, 4(1):14–31, feb 2005.

[40] Anwar I. Elwalid and Debasis Mitra. Fluid models for the analysis and design of statistical multiplexing with loss priorities on multiple classes of bursty traffic. In *Proc. of IEEE INFOCOM*, pages 415–425, 1992.

[41] E. A. Emerson. *Temporal and modal logic*. MIT Press, 1990.

[42] EmStar. Software for Wireless Sensor Networks, `cvs.cens.ucla.edu/emstar/`.

[43] EmuLab. Total network testbed, `www.emulab.net/`.

[44] D. Engler and M. Musuvathi. Static analysis versus software model checking for bug finding. In *Proc. 5th International Conference on Verification, Model checking and Abstract Interpretation*, volume 2937, pages 191–210, Venice, Italy, January 2004.

[45] Ansgar Fehnker, Matthias Fruth, and Annabelle McIver. Graphical modelling for simulation and formal analysis of wireless network protocols. In *Proceedings of the Workshop on Methods, Models and Tools for Fault Tolerance (MeMoT 2007) at the 7th International Conference on Integrated Formal Methods (IFM 2007)*, pages 80–87, July 2007.

[46] M. J. Fernández-Iglesias, J. C. Burguillo-Rial, no F. J. González-Casta and M. Llamas-Nistal. Wireless protocol testing and validation supported by formal methods: a hands-on report. *Journal of Systems and Software*, 75(1-2):139–154, 2005.

[47] M. Garetto, P. Giaccone, and E. Leonardi. Capacity scaling in delay tolerant networks with heterogeneous mobile nodes. In *Proc. of ACM MobiHoc*, pages 15–17, 2007.

[48] M. Garetto, P. Giaccone, and E. Leonardi. Capacity scaling of sparse mobile ad hoc networks. In *Proc. of IEEE INFOCOM*, pages 206–210, 2008.

[49] M. Garetto, T. Salonidis, and E. Knightly. Modeling per-flow throughput and capturing starvation in csma multi-hop wireless networks. In *Proc. of IEEE INFOCOM*, pages 1–13, 2006.

[50] M. Garetto, T. Salonidis, and E. W. Knightly. Modeling per-flow throughput and capturing starvation in csma multi-hop wireless networks. *IEEE/ACM Trans. Netw.*, 16(4):864–877, 2008.

[51] GENI. Global Environment Network Innovations, `www.geni.net/`.

[52] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. Emstar: a software environment for developing and deploying wireless sensor networks. In *Proc. of USENIX General Track*, 2004.

[53] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer. A system for simulation, emulation, and deployment of heterogeneous sensor networks. In *Proc. of SenSys*, April 2004.

[54] Uwe Glässer and Qian-Ping Gu. Formal description and analysis of a distributed location service for mobile ad hoc networks. *Theoretical Computer Science*, 336(2-3):285–309, 2005.

[55] GloMoSim. Global Mobile information systems Simulator, `pcl.cs.ucla.edu/projects/glomosim/`.

[56] K. Godary, I. Augé-Blum, and A. Mignotte. Sdl and timed petri nets versus uppaal for the validation of embedded architecture in automative. In *Proc. 1st Forum on Specification and Design Languages (FDL)*, Lille, France, Sep 2004.

[57] P. Godefroid. Model checking for programming languages using verisoft. In *Principles of Programming Languages*, pages 174–186, Paris, France, January 1997. ACM Press.

[58] M. Grossglauser and D. Tse. Mobility increases the capacity of ad-hoc wireless networks. *IEEE/ACM Transaction on Networking*, 10(4):477–486, August 2002.

[59] E. Gunter and D. Peled. Model checking, testing and verification working together. *Formal Aspects of Computing*, 17(2):201–221, August 2005.

[60] K. Havelund and T. Pressburger. Model checking java programs using java pathfinder. *Software Tools for Technology Transfer*, 2(4):366–381, 2000.

[61] E. Heck. *Performance Evaluation of Formally Specified Systems - the intergration of SDL with HIT*. PhD thesis, University of Dortmund, 1996.

[62] Leah Hoffman. In search of dependable design. *Communications of the ACM*, 51(7):14–16, July 2008.

[63] HOL, 2005. http://www.cl.cam.ac.uk/Research/HVG/HOL/.

[64] G. J. Holzmann. *Design and Validation of Computer Networks*. Prentice-Hall International Editions, AT&T Bell Laboratories, 1991.

[65] G. J. Holzmann. *The SPIN Model Checker*. Addison-Wesley, 2003.

[66] ITU-T. *Message Sequence Charts (MSC'96)*, 1996.

[67] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999.

[68] Java-Sim. `www.j-sim.org/`.

[69] E. Javier and P. Raymond. The lucky language reference manual. Technical Report TR-2004-06, VERIMAG, 2005.

[70] Ron Jeffries and Scott W. Ambler. *Agile Modeling*. Jon Wiley & Sons, 2002.

[71] V. Jones, A. Rensink, T. Ruys, E. Brinksma, and A. van Halteren. A formal mda approach for mobile health systems. In *Proc. of the 2nd European Workshop on Model Driven Architecture with an emphasis on Methodologies and Transformations,*, pages 28–35, Canterbury, England, September 2004.

[72] S. Keshav. Real:a network simulator. Technical report, Technical Report 88/472. University of California, Berkeley, 1988.

[73] W. Kiess and M. Mauve. A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Networks*, 5(3), 2007.

[74] Wolfgang Kiess and Martin Mauvea. A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Networks, Elsevier*, 5(3):538–576, April 2007.

[75] M. Kim, M. Viswanathan, H. Ben-Abdallah, S. Kannan, I. Lee, and O. Sokolsky. Formally specified monitoring of temporal properties. In *Proc. 11th Euromicro Conference on Real-Time Systems*, pages 1–14, New York, USA, June 1999.

[76] E. Kindler. Safety and liveness properties: A survey. *Bulletin of the European Association for Theoretical Computer Science*, (53):268–272, Jun 1994.

[77] Leonard Kleinrock. *Queueing Systems*. Wiley-Interscience, 1975.

[78] M. Kropff, T. Krop, M. Hollick, P. S. Mogre, and R. Steinmetz. A survey on real world and emulation testbeds for mobile ad hoc networks. In *Proc. 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM)*, March 2006.

[79] S. Kurkowski, T. Camp, and M. Colagrosso. Manet simulation studies: the incredibles. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(4):50–61, October 2005.

[80] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilitic model checking of the ieee 802.11 wireless local area network protocol. In *Proc. 2nd Joint International Workshop on Process Algebra and Probabilitics Methods, Performance Modeling and Verification*, pages 169–187, Copenhagen, Denmark, Jul 2002.

[81] Seok Myun Kwon and Jin Suk Kim. Coverage ratio in the wireless sensor networks using monte carlo simulation. In *Proc. of the 4th International Conference on Networked Computing and Advanced Information Management*, pages 235–238, Gyeongju, Korea, 2008.

[82] S. K. Lahiri. *Unbounded System Verification Using Decision Procedure and Predicate Abstraction*. PhD thesis, Carnegie Mellon University, September 2004.

[83] Shuvendu K. Lahiri and Randal E. Bryant. Predicate abstraction with indexed predicates. *ACM Transaction on Computational Logic*, 9(1):4, 2007.

[84] K. G. Larsen, P. Petterson, and W. Yi. Uppaal in a nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(2):134–152, October 1997.

[85] T. Lin, S. Midkiff, and J. Park. A framework for wireless ad hoc routing protocols. In *Proc. of IEEE Wireless Communications and Networking Conf. (WCNC)*, pages 1162–1167, New Orleans, Louisiana, USA, March 2003.

[86] H. Lundgren, E. Nordström, and C. Tschudin. Coping with communication gray zones in ieee 802.11b based ad hoc networks. In *Proc. of the 5th ACM international workshop on Wireless Mobile Multimedia*, pages 49–55, Atlanta, Georgia, USA, 2002.

[87] S. Maag, C. Grepet, and A. Cavalli. A formal validation methodology for manet routing protocols based on nodes' self similarity. *Computer Communincations*, 31(4):827–841, 2008.

[88] S. Maag and F. Zaidi. Testing methodology for an ad hoc routing protocol. In *Proc. of the ACM international workshop on Performance monitoring, measurement, and evaluation of heterogeneous wireless and wired networks*, pages 48–55, Terromolinos, Spain, 2006.

[89] S. Maag and F. Zaidi. A step-wise validation approach for a wireless routing protocol. *Posts, Telecommunications and Information Technology Journal*, 1:34–40, 2007.

[90] J. P. Macker, W. Chao, and J. W. Weston. A low-cost, ip-based mobile network emulator (mne). Technical report, Naval Research Lab Washington DC, ADA464904, 2003.

[91] Louis Mandel and Marc Pouzet. ReactiveML, a reactive extension to ML. In *Proc. 7th ACM SIGPLAN International conference on Principles and Practice of Declarative Programming (PPDP'05)*, Lisbon, Portugal, July 2005.

[92] Zohar Manna, Nikolaj S. Bjørner, Anca Browne, Michael Colón, Bernd Finkbeiner, Mark Pichora, Henny B. Sipma, and Tomás E. Uribe. An update on STeP: Deductive-algorithmic verification of reactive systems. In Rudolf Berghammer and Yassine Lakhnech, editors, *Tool Support for System Specification, Development and Verification*, Advances in Computing Science, pages 174–188. Springer-Verlag, 1999.

[93] A.A. Markov. *Extension of the limit theorems of probability theory to a sum of variables connected in a chain. Markov Chains*. John Wiley and Sons, 1971.

[94] J. Martin. *RAD, Rapid Application Development*. MacMillan Publishing Co, New York, 1990.

[95] A. McIver. Quantitative mu-calculus analysis of power management in wireless networks. In *Proc. 3rd International Theoretical Aspects of Computing*, pages 50–64, Tunis, Tunisia, 2006. Springer.

[96] A. K. McIver and A. Fehnker. Formal techniques for the analysis of wireless networks. In *Proc. 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, pages 263–270, November 2006.

[97] Koceilah Merouane, Cyril Grepet, and Stephane Maag. A methodology for interoperability testing of a manet routing protocol. In *Proc. of the 3rd International Conference on Wireless and Mobile Communications*, Washington, DC, USA, 2007. IEEE Computer Society.

[98] A. Mitschele-Thiel and B. Müller-Clostermann. Performance engineering of sdl/msc systems. *Computer Networks and ISDN Systems*, 31(17):1801–1816, 1999. Elsevier.

[99] Netkit network emulator. `www.netkit.org/features.html`.

[100] www.isi.edu/nsnam/ns.

[101] OneLab European platform. `www.one-lab.org/wiki/view/OneLab`.

[102] OPNET simulator. `www.opnet.com/`.

[103] ORBIT. Radio grid testbed, `www.orbit-lab.org/`.

[104] Other testbeds. `www.emulab.net/docwrapper.php3?docname=otheremulabs.html`.

[105] PEPA. `http://www.dcs.ed.ac.uk/pepa`.

[106] C. Perkins, E. Belding-Royer, and S. Das. Rfc 3561, ad hoc on-demand distance vector (aodv) routing. Technical report, IETF, 2003.

[107] James Lyle Peterson. Petri nets. *ACM Computing Surveys*, 9(3):223–252, 1977.

[108] James Lyle Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall, 1981.

[109] Carl A. Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, 1962.

[110] D. Petriu and M. Woodside. Software performance models from systems scenarios in use case maps. In *12th Int. Performance Evaluation: Modelling Techniques and Tools*, pages 1–8, London, UK, April 2002. Springer Verlag.

[111] M. Pizzonia and M. Rimondini. Easy emulation of complex networks on inexpensive hardware. In *Proc. 4th International Conf. on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM)*, March 2008.

[112] PlanetLab Europe platform. `www.planet-lab.eu/`.

[113] PlanetLab platform. `www.planet-lab.org/`.

[114] A. Pnueli. A temporal logic of programs. *Theoretical Computer Science*, 13(1):45–60, 1981.

[115] M. Puzar and T. Plagemann. Neman: A network emulator for mobile ad-hoc networks. Technical report, Department of Informatcis, University of Oslo. TR321, ISBN 82-7368-274-9, 2005.

[116] S. Ray, D. Starobinski, and J. B. Carruthers. Performance of wireless networks with hidden nodes: a queuing-theoretic analysis. *Journal of Computer Communications (Special Issue on the Performance issues of Wireless LANs, PANs and ad hoc networks*, 28(10):1179–1192, 2005.

[117] M. Rimondini. Emulation of computer networks with netkit. Technical report, TR RT-DIA-113-2007, University of Roma Tre, 2007.

[118] RON. Resilient Overlay Networks, `nms.csail.mit.edu/ron/`.

[119] W. W. Royce. Managing the development of large software systems: concepts and techniques. In *Proc. of International Conference on Software Engineering Table of Contents*, pages 328–338, Monterey, California, USA, 1987.

[120] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo Method*. John Wiley and Sons, 2007.

[121] J. Rutten, M.Kwiatkowska, G. Norman, and D. Parker. Mathematical techniques for analysing concurrent and probabilitics systems. *CRM Monograph Series*, 23, March 2004. American Mathematical Society.

[122] S. K. Lahiri and R. E. Bryant. Uclid-pa homepage: Verification tool for uclid models using predicate abstraction, 2005.

[123] Ludovic Samper, Florence Maraninchi, Laurent Mounier, and Louis Mandel. Glonemo: Global and accurate formal models for the analysis of ad-hoc sensor networks. In *Proc. 1st International Conference on Integrated Internet Ad hoc and Sensor Networks*, Nice, France, May 2006.

[124] SENS. A Sensor, Environment and Network Simulator, `osl.cs.uiuc.edu/sens/`.

[125] SENSE. Sensor Network Simulator and Emulator, `www.ita.cs.rpi.edu/sense/index.html`.

[126] SensorSim. A Simulation Framework for Sensor Networks, `nesl.ee.ucla.edu/projects/sensorsim/`.

[127] M. Sgroi, J. L. da Silva, F. De Bernardinis, F. Burghardt, A. Sangiovanni-Vincentelli, and J. Rabaey. Designing wireless protocols: Methodology and applications. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3726–3729, June 2000.

[128] SimReal. SimReal network simulator, `nsl10.csie.nctu.edu.tw/`.

[129] J. M. Spivey and M. Spivey. *An Introduction to Logic Programming Through Prolog*. Prentice Hall, 1996.

[130] M. Steppler and M. Lott. Speet - sdl performance evaluation tool. In *SDL'97*, pages 53–67. Elsevier, 1997.

[131] L. Sterling and E. Shapiro. *The Art of Prolog*. The MIT Press, 1994.

[132] Cheng Lin Tan and Stephen Pink. Mobicast: a multicast scheme for wireless networks. volume 5, pages 259–271, Hingham, MA, USA, 2000. Kluwer Academic Publishers.

[133] T.Clausen and P.Jacquet. *Optimized Link State Routing Protocol (OLSR) - RFC3626*. INRIA, ietf edition, October 2003.

[134] O. Tickoo and B. Sikdar. Queueing analysis and delay mitigation in ieee 802.11 random access mac based wireless networks. In *Proc. of IEEE INFOCOM*, pages 7–11, 2004.

[135] TIDIA. TIDIA/Kyatera Emulab testbed, `www.emulab.larc.usp.br/index`.

[136] TinyOS Simulator. `www.cs.berkeley.edu/ pal/research/ tossim.html`.

[137] Kishor S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. John Wiley and Sons, 2001.

[138] TWISC. Network emulation testbed, `testbed.ncku.edu.tw/`.

[139] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, Duke University, 2000.

[140] Verilog. *ObjectGEODE Simulator*, 1997.

[141] H. Van Vliet. *Sotware Engineering: Principles and Practice (2nd Edition)*. Wiley, 1999.

[142] Ying Wang and Mudi Xiong. Monte carlo simulation of leach protocol for wireless sensor networks. In *Proc. of the 6th International Conference on Parallel and Distributed Computing Applications and Technologies*, pages 85–88, Washington, DC, USA, 2005.

[143] T. Watteyne, I. Auge-Blum, and S. Ubeda. Formal qos validation approach on a real-time mac protocol for wireless sensor networks. Technical Report 5782, RR-5782, INRIA, December 2005.

[144] M. Wei, F. Dubois, D. Vincent, and P. Combes. Looking for better integration of design and performance engineering. In Springer, editor, *SDL 2003 : System design*, pages 1–17, Stuttgart, Germany, jul 2003.

[145] O. Wibling. Ad hoc routing protocol validation. Master's thesis, Uppsala University, 2005.

[146] WSim. WSim simulator, `worldsens.citi.insa-lyon.fr/joomla/index.php`.

[147] WSNet. Wireless Sensor Network simulator, `worldsens.citi.insa-lyon.fr/joomla/index.php`.

[148] X-Bone project. `www.isi.edu/x-bone/`.

[149] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proc. of ACM SenSys*, pages 13–24, 2004.

[150] F. Zaïdi, A. Cavalli, and E. Bayse. NetworkProtocol Interoperability Testing based on Contextual. In *24th Annual ACM Symposium on Applied Computing*, pages 321–327, Hawaii, USA, March 2009. ACM. TO be published.

[151] I. Zakkuidin, T. Hawkins, and N. Moffat. Towards a game theoretic understanding of ad hoc routing. *Electronic Notes in Theoretical Computer Science*, 119(1):67–92, February 2005.

Contents

1 Introduction **3**

2 Protocol engineering **5**

3 The WSONs challenges **9**

4 Non-formal approaches **10**
4.1 Simulation 10
4.2 Emulation 12
4.3 Testbeds 13
4.4 Prototyping 15

5 Formal approaches **15**
5.1 System modeling 16
5.2 Verification techniques 18
5.2.1 Model checking approaches 19
5.2.2 Deductive verification approaches 21
5.2.3 Hybrid approaches 22
5.3 Formal testing techniques 22
5.4 Hybrid techniques 25

6 Objectives, limitations, and drawbacks **25**
6.1 Regarding non-formal approaches 25
6.2 Regarding formal approaches 27

7 Domains convergence **29**
7.1 Convergence at the modeling step 30
7.2 Convergence at validation level 31
7.3 Ways for converging 32

8 Conclusion **33**

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 Orsay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399